**Technical Design Document Exercise 1**

**Name:** Logan Flynn

**Date Created:** 08/31/2025

**Program Description:** This program simulates a limited inventory cinema ticket presale. Buyers are offered tickets until they sell out. Each buyer can purchase up to a per buyer amount limit, but not more than the amount that remains. The program validates the buyer's input, supports skipping a buyer by pressing enter, and prints the amount of tickets remaining after each purchase.

**Functions used in the Program (list in order as they are called):**

### **1. Function Name:** prompt\_ticket\_request(remaining: int) -> int | None

**Description:** Prompts a buyer for how many tickets they want. Enforces that the quantity is a whole number between 1 and min(MAX\_PER\_BUYER, remaining). If the user presses Enter, the buyer is skipped and the function returns None.

**Parameters:** remaining (int) — number of tickets still available.

**Variables:**

1. limit (int) - the maximum allowed for the current buyer (the smaller of MAX\_PER\_BUYER and remaining).
2. raw (str) - raw input from the user.
3. qty (int) - parsed, validated quantity.

**Logical Steps:**

1. Compute limit = min(MAX\_PER\_BUYER, remaining).
2. Loop until valid input is received:  
   a. Prompt: “How many tickets would you like (1–{limit})? Press Enter to skip.”  
   b. If input is empty - return None.c. If input is not a whole number - print error and continue.  
   d. Convert to qty.  
   e. If qty < 1, qty > MAX\_PER\_BUYER, or qty > remaining - print specific message and continue.
3. Return the valid qty.

**Returns:** int | None - a valid quantity, or None if the buyer skips.

### **2. Function Name:** process\_purchase(remaining: int, qty: int | None) -> tuple[int, bool]

**Description:** Processes a single buyer’s decision. If qty is None, no purchase occurs. Otherwise subtracts qty from remaining and prints a confirmation showing the updated number of tickets left.

**Parameters:**

remaining (int) - tickets available before this buyer.

qty (int | None) - requested quantity or None if skipped.

**Variables:**

1. new\_remaining (int) - updated ticket count after purchase.

**Logical Steps:**

1. If qty is None - return (remaining, False).
2. Compute new\_remaining = remaining - qty.
3. Print: “Purchase successful…” with the new remaining count.
4. Return (new\_remaining, True).

**Returns:** tuple[int, bool] - (new\_remaining, purchased\_flag).

### **3. Function Name:** main() -> None

**Description:** Coordinates the presale. Greets the user, initializes counters, repeatedly offers tickets to buyers until sold out, and finally prints a summary.

**Parameters:** None

**Variables:**

1. remaining (int) - starts at MAX\_TICKETS; decremented by purchases.
2. total\_buyers (int) - counts buyers who successfully purchased.
3. qty (int | None) - request returned from prompt\_ticket\_request.
4. new\_remaining (int) - updated count returned from process\_purchase.
5. purchased (bool) - whether a purchase occurred for this buyer.

**Logical Steps:**

1. Print banner and rules (total inventory and per-buyer limit).
2. Set remaining = MAX\_TICKETS, total\_buyers = 0.
3. While remaining > 0:  
   a. qty = prompt\_ticket\_request(remaining)  
   b. (new\_remaining, purchased) = process\_purchase(remaining, qty)  
   c. If purchased:
   * total\_buyers += 1
   * remaining = new\_remaining  
     d. Else: print “No purchase made…” and continue to next buyer.
4. If remaining == 0 print “SOLD OUT”, else “PRESALE ENDED”.
5. Print total\_buyers.

**Returns:** None

## **Overall Logical Steps**

1. Call main().
2. Inside main, repeatedly call prompt\_ticket\_request and process\_purchase until the tickets are gone or buyers skip.
3. Print final status and total number of purchasing buyers.

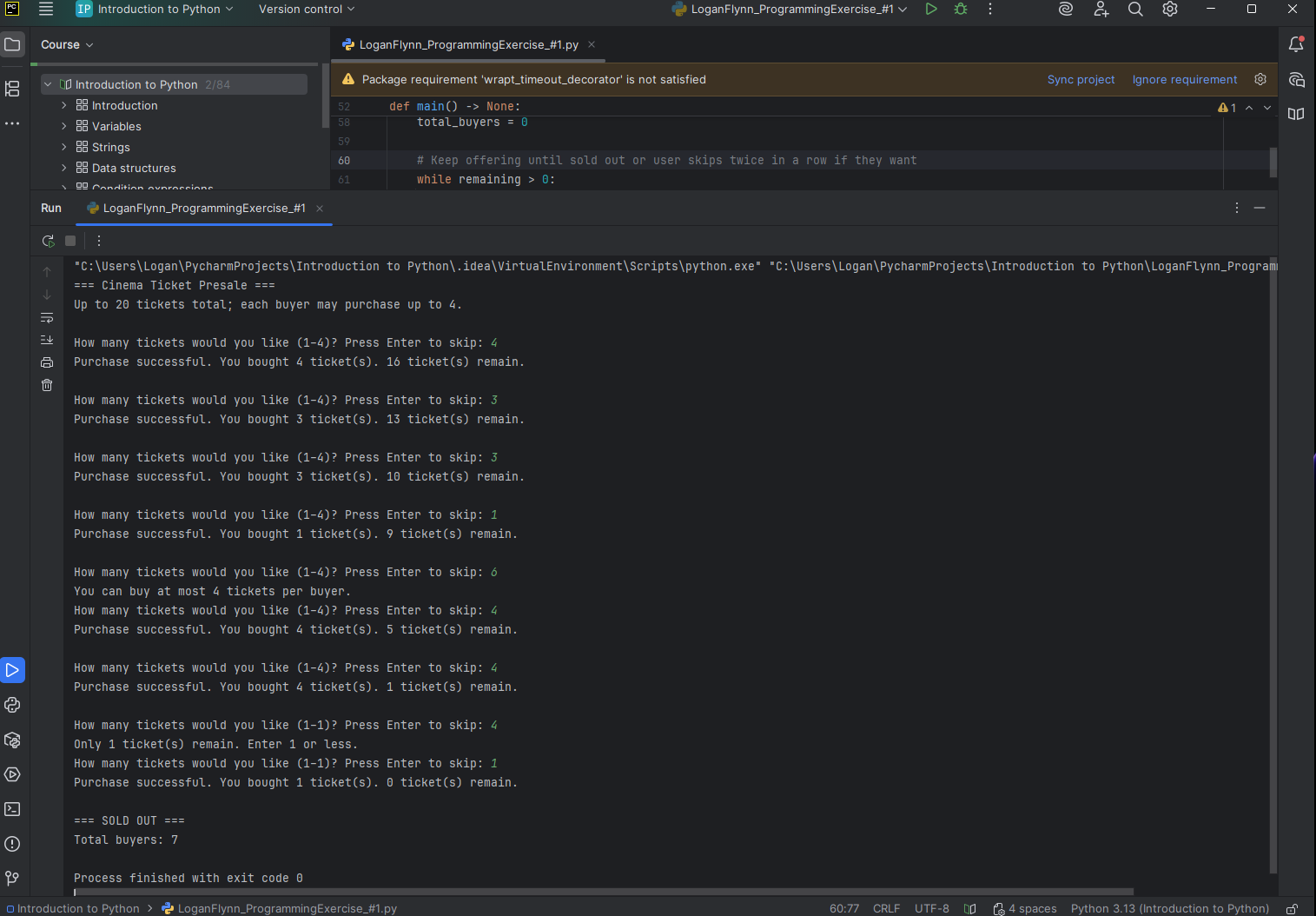
## **Constants**

* MAX\_TICKETS = 20
* MAX\_PER\_BUYER = 4

**Link to your repository:**

[loganflynnn (Logan Flynn)](https://github.com/loganflynnn)

**Output Screenshot:**

****